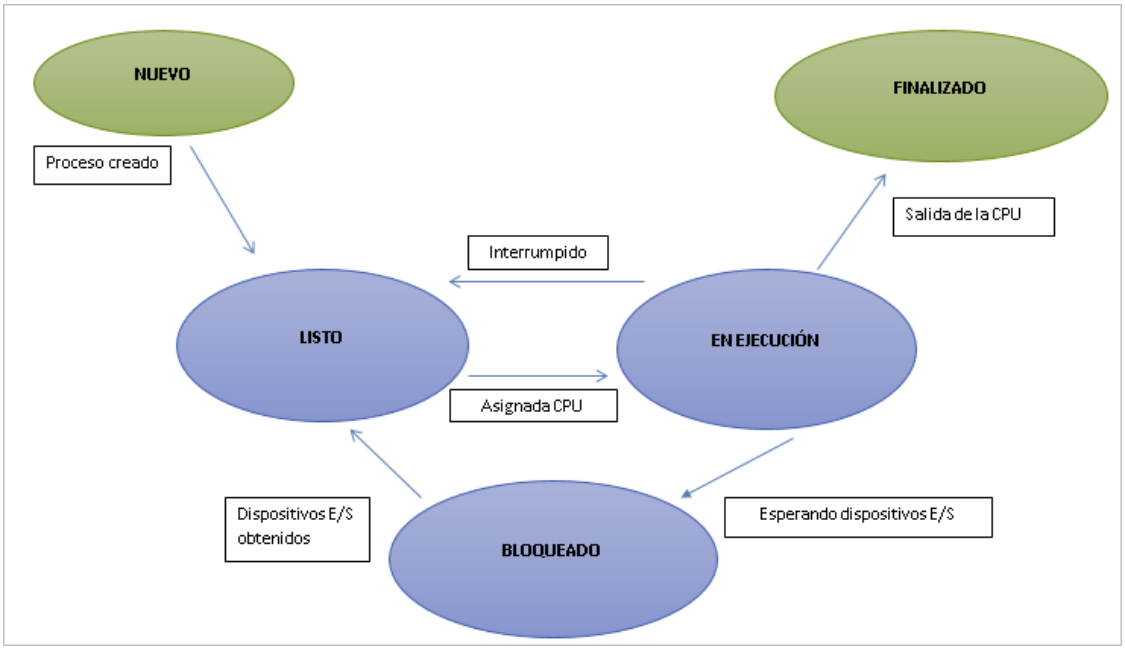
Funciones de los Sistemas Operativos

- Gestión de procesos y planificación de la CPU
- Gestión de memoria
- Gestión de archivos ó ficheros
- Gestión de recursos ó periféricos
- Gestión de usuarios y permisos

Gestión de Procesos y planificación de la CPU

Un proceso es un programa en ejecución
Estados de un proceso:

- Ejecución, en posesión de la CPU
- Listo o preparado, esperando turno para entrar en la CPU
- Bloqueado o en espera, esperando conseguir un recurso.



Planificación de la CPU: consiste en repartir el tiempo de CPU entre los procesos para aumentar el rendimiento, siendo el planificador o scheduler el encargado de decidir en qué orden se debe atender las solicitudes de uso del procesador.

Algoritmos de planificación de CPU

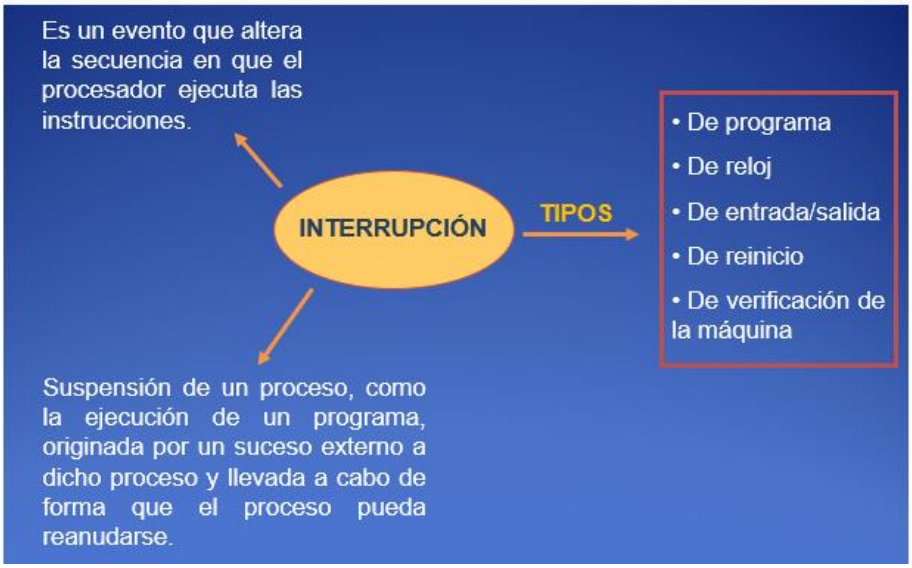
- FIFO – primero en entrar – primero en salir.
  PROCESO1 luego PROCESO2 luego PROCESO3
- SJF (shortest Job First) – primero la tarea más corta.
- ROUND ROBIN ó planificación circular, más sencillo, justo y más utilizado. Utiliza un tiempo quantum, NO se establecen prioridades. Al finalizar el quantum, se desaloja de la CPU y pasa a la cola de listos.
- Otros tipos de algoritmos están basados en prioridades, el de > prioridad quita de la CPU al de < prioridad y luego continúa éste. Ante igual prioridad se procesan los más cortos.

Las interrupciones

Ante una situación particular, la CPU suspende temporalmente el proceso que estaba realizando (se produce la **interrupción** y pasa a dedicarse seguidamente a una tarea independiente y prioritaria necesaria para resolver la situación, guardando toda la información necesaria para regresar y reanudar la tarea que estaba realizando.

El gestor de interrupciones se encarga de gestionar 5 tipos de interrupciones:

- ✓ Interrupciones por errores de máquina (p.e. fallo de CPU ó memoria).
- ✓ Errores de programa
- ✓ Errores con dispositivos
- ✓ De reinicio
- ✓ De reloj



https://es.wikipedia.org/wiki/Interrupci%C3%B3n

Sincronización de procesos y tareas. Los semáforos

Los procesos pueden ser **independientes** o **cooperantes**

INDEPENDIENTES COOPERANTES

En la multitarea ó multiprogramación es necesaria la sincronización de procesos y tareas, de esta función se encargan los programas de gestión de semáforos, los cuales gestionan el tráfico de procesos que utilizan datos compartidos.

En caso de procesos cooperantes, cuando 2 procesos comparten un recurso, es necesario prohibir que uno lea y otro escriba a la vez, intentando conseguir:

Exclusión Mutua: Mecanismo que consiste en que un solo proceso excluye temporalmente a todos los demás para usar un recurso compartido de forma que garantice la integridad del sistema.

Sección Crítica: sólo una parte del proceso que tengan en común, se produce la exclusión mutua.

Gestión de Memoria

Cada proceso en ejecución necesita un área de memoria para ubicar los datos, las instrucciones y la pila (zona de trabajo durante la ejecución).
El gestor de memoria es la parte del S.O. encargado de asignar memoria a los procesos y gestionar su uso. Debe proporcionar protección y uso compartido, es decir, debe proporcionar un espacio de memoria para cada proceso que lo necesite y controlar que ningún proceso trabaje en zonas de memoria que no le corresponden.

La memoria se puede asemejar a una tabla dividida en celdas de igual tamaño. En cada celda se alberga un dato y cada celda tiene una dirección. El tamaño de la celda varía de una arquitectura a otra, puede ser un bit, un byte, una palabra (8 bits, 16 bits, 32 bits, 64 bits).

Funciones principales del gestor de memoria:

- Ofrecer a cada proceso un espacio lógico de memoria.
- Proporcionar protección entre procesos.
- Permitir que los procesos se compartan la memoria.
- Dar soporte a las distintas regiones del proceso.

Los criterios para evaluar un gestor de memoria son:

- Memoria desaprovechada: es la memoria que se pierde en el proceso de asignación.
- Complejidad en el tiempo: es el tiempo perdido en el proceso de acceso a la zona de memoria asignada.
- Procesos complementarios de accesos a memoria. Son los pasos a realizar para acceder a un dato en la memoria.

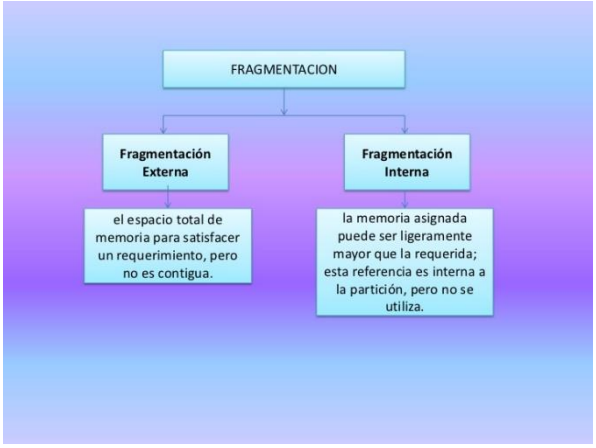
Existen algoritmos para la gestión de memoria:

Particiones estáticas o fijas: es la forma más sencilla consiste en dividir la memoria en X partes fijas de igual tamaño. Si es muy pequeña, necesitará más, y si es muy grande, se desaprovechará la memoria.

Particiones dinámicas o variables: esta técnica consiste en dividir la memoria de forma dinámica, según se vaya necesitando, tiene la desventaja que produce huecos en la memoria y necesita compactación para desplazar los procesos que estén contiguos, de forma que toda la memoria libre quede junta en un bloque.

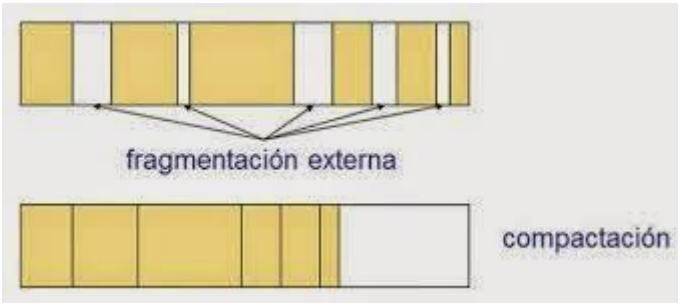
Fragmentación

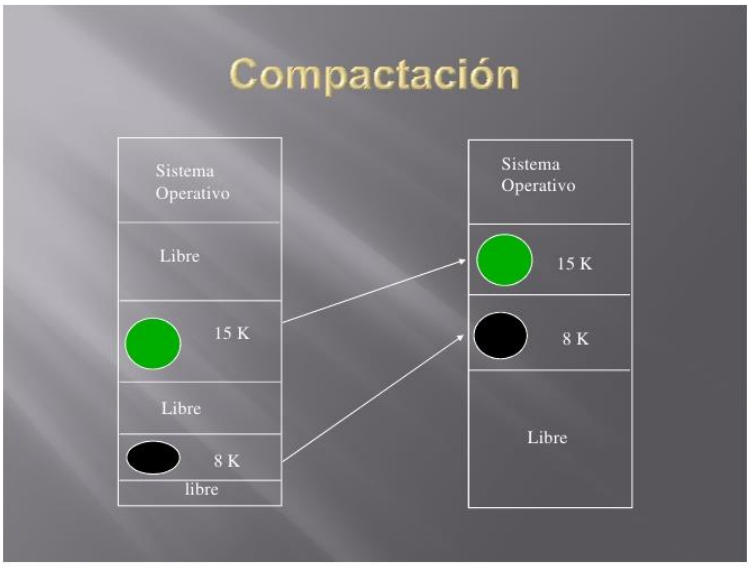
- **Fragmentación Interna,** los bloques de memoria son más grandes que lo asignado, por tanto, queda memoria sin utilizar, desperdiciada.
- **Fragmentación Externa,** la memoria se divide en bloques variables, se desaprovecha el espacio entre bloques, p.e. 1 proceso de 30KB no podría entrar en 2 bloques de 20KB no contiguos, el proceso no se ejecutaría por falta de memoria.

- 

En general el problema de la fragmentación externa, sería necesario poder reubicar programas en tiempo de ejecución. Cuando éste proceso termine, el espacio generado produciría más fragmentación, por ello existen técnicas de compactación que solucionan el problema.

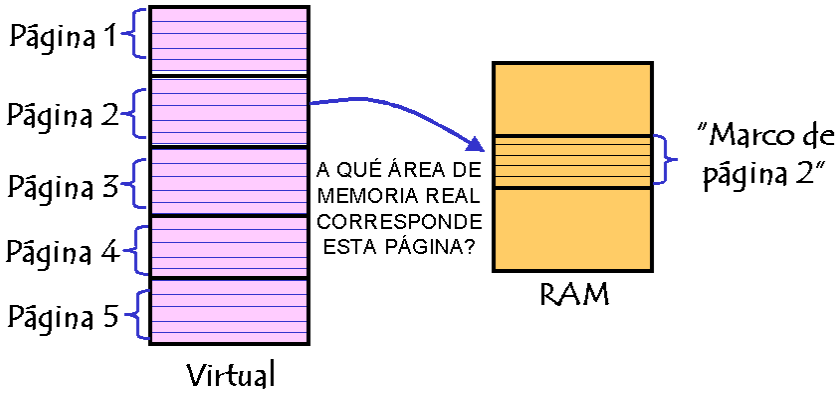
Técnicas de compactación: consisten en desplazar los programas para que todo el espacio libre quede contiguo.



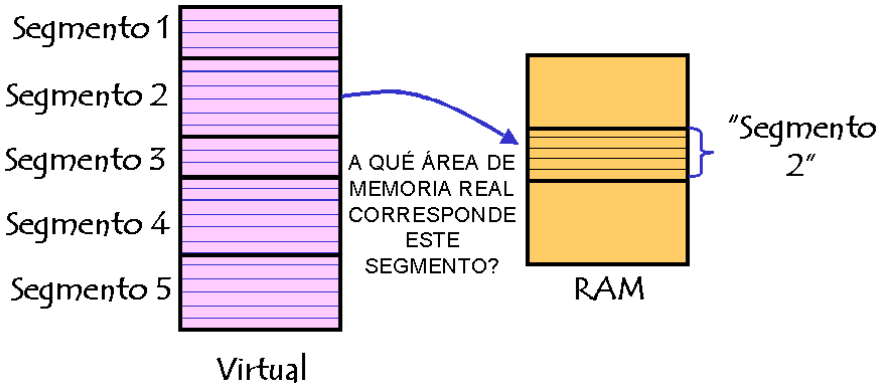


Otras técnicas de gestión de memoria:

La **paginación** consiste el dividir el espacio de memoria necesario en trozos pequeños llamados **frames ó marcos**, lo mismo con el programa, cuyos trozos se llamados **páginas.** Solo se carga en memoria la página que se necesita. Si la página que solicita no está cargada en memoria se produce un **fallo de página** y luego se carga la página solicitada. Esta técnica elimina la fragmentación externa, pero no la fragmentación interna.



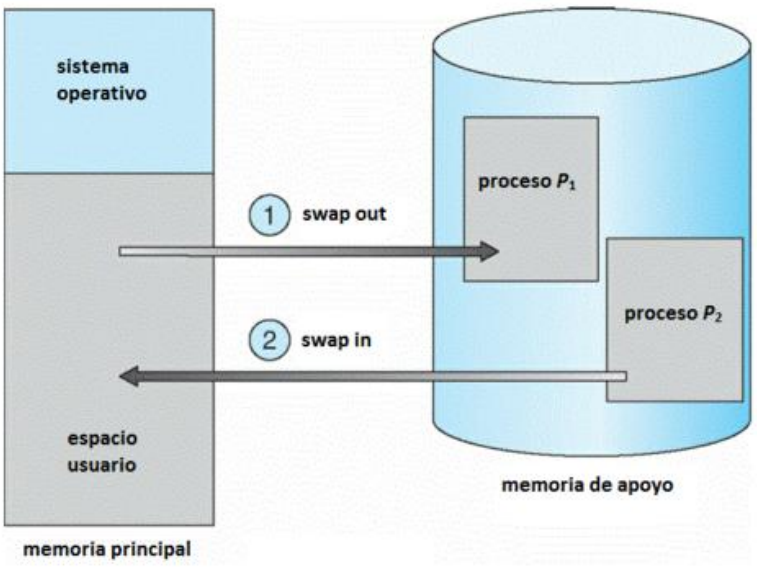
La **segmentación** consiste en dividir el programa en segmentos de longitud variable que, junto a la paginación, realizan una gestión más óptima de la memoria. Estos segmentos pueden ser funciones o procedimientos que pueden ser compartidos en memoria. Esta técnica es similar a la de partición dinámica.



La **memoria virtual** permite ejecutar programas más grandes que la memoria RAM, utilizando el almacenamiento secundario como si fuera parte de la memoria principal, la limitación estaría en la capacidad del disco duro. Para la gestión de la memoria virtual, en Windows, se recomienda asignar el 1,5 del tamaño de total de la RAM, y si hay suficiente espacio en disco, entonces aplica el 2 * RAM.
Esta técnica consiste en guardar en disco (memoria auxiliar ó secundaria) los bloques de datos que no necesita y subir a memoria principal, el bloque que necesita.

La paginación difiere de la segmentación en que las páginas son de tamaño fijo y los segmentos no. El uso de las técnicas de paginación o segmentación dependerá del SO utilizado y de la máquina en la que se empleen, además de las necesidades del software.

Swapping o intercambio: es similar a la memoria virtual. Cuando varios usuarios están ejecutando procesos en un mismo ordenador, éste se ve obligado a cargarlos en RAM. Según el estado en el que se encuentre el proceso de cada usuario, la memoria se irá liberando de su proceso, pasándolo temporalmente (intercambiado) a almacenamiento secundario. De esta forma, la memoria interna queda liberada para que en ella se pueda almacenar otro proceso del mismo usuario o de otro.
Si un usuario vuelve a solicitar su proceso para seguir ejecutándolo, se produce el denominado **swap-in** que consiste en pasar el programa de la zona secundaria a la memoria interna.
La zona secundaria de intercambio es un área independiente del sistema de ficheros que permite un acceso rápido a los datos.



Procesos reubicables, reentrantes, residentes y reutilizables:
1. Los procesos **reubicables,** son los que una vez cargados en RAM, pueden cambiar de ubicación.
2. Los **reentrantes**, si no se están ejecutando, dejan la memoria libre para otros procesos. Suelen ser los procesos gestionados por la memoria virtual.
3. Los procesos **residentes**, son los que una vez cargados en memoria, permanecerán en ella hasta que se apague el ordenador. No cambian nunca de ubicación, suelen ser programas antivirus, de análisis del sistema, de monitorización.
4. Los **reutilizables**, procesos utilizados por varios usuarios a la vez.

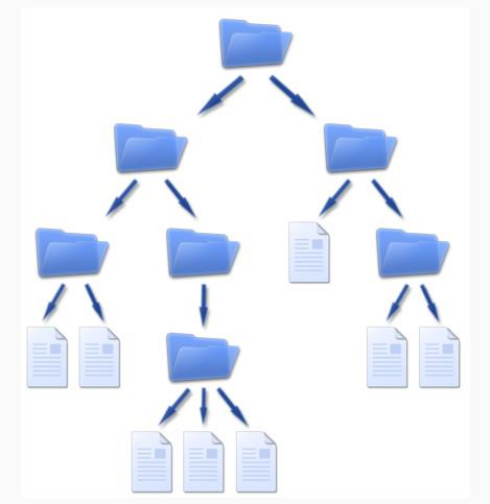
Gestión de Ficheros ó Archivos

Los archivos y programas de los usuarios se guardan, en dispositivos de almacenamiento, con un nombre, un formato, se establecen permisos de acceso, propietario, etc. Una vez guardados, se pueden abrir, borrar, cambiar de nombre, modificar, etc. Todas las operaciones que se pueden realizar con los ficheros son funciones que proporcionan los sistemas operativos.

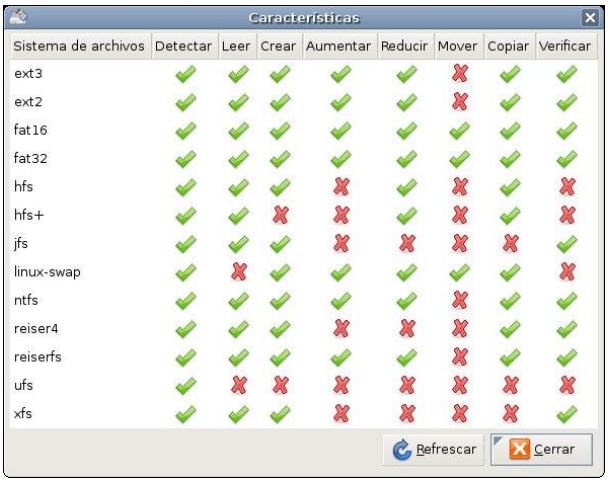
Además, definen unas estructuras para que se puedan almacenar en los diferentes dispositivos de almacenamiento. Estas estructuras son distintas de unos a otros sistemas operativos, a veces compatibles entre los distintos S.O. y otras veces, totalmente incompatibles.

La gestión de ficheros, incluye la asignación de ubicación física y lógica a los ficheros. La lectura y escritura de un fichero (de cualquier tipo) se realiza a través de un bloque, que es la unidad de acceso a ficheros. El bloque es un conjunto de posiciones del disco lógicamente contiguos, y su tamaño dependerá del gestor de ficheros.

El gestor de ficheros, debe gestionar el almacenamiento auxiliar, el cual es organizado de una forma determinada. A esta organización se denomina **sistema de ficheros**.



Existen técnicas de compactación o desfragmentación de almacenamiento secundario o auxiliar, que reorganiza el espacio libre, como formas de mejorar el rendimiento del equipo, como resultado maximizará la lectura y escritura del HD.



Gestión de Recursos o Periféricos

Control de los periféricos de I/O del ordenador, acceder a ellos de forma controlada, gestionar los errores ocasionados por los dispositivos, gestionar prioridades, asigna y libera los dispositivos y planifica la E/S. Cada periférico está compuesto por un componente hardware, otro software (driver o device driver) y algunos, por firmware (controladora o device controller)

Gestión de Usuarios y permisos

Se encarga de la gestión de usuarios, cuentas (altas, bloqueos, inhabilitación, perfiles, etc..) y contraseñas (directivas, cambios, etc..), de grupos de usuarios, de permisos sobre recursos, cuotas de disco, accesos a ficheros, y el control de acceso al sistema. También registra los sucesos y eventos, que suceden en todo momento, realiza auditorías, estadísticas, monitoreos, etc.

Software de aplicación, de servicio y utilidades.

Software de aplicación:

- *Aplicaciones de Uso General:*
  - Procesadores de texto, hojas de cálculo, gestores de Bases de Datos, Aplicaciones de Diseño Gráfico, etc.
- *Aplicaciones de Uso específico:*
  - Contabilidad, Facturación, CAD (diseño asistido por computador), gestión de hospitales, etc.

Software de Servicio:

- Programas traductores (permiten obtener lenguaje de máquina ó binario)
  - Ensambladores
  - Intérpretes
  - Compiladores

Tipos de Lenguajes de programación de aplicaciones.

- Lenguajes de bajo nivel: L. ensamblador y lenguaje de máquina.
- Lenguajes de alto nivel: C, C++, VB.Net, Cobol, Ada, Fortran, Pascal, Python, java, etc.

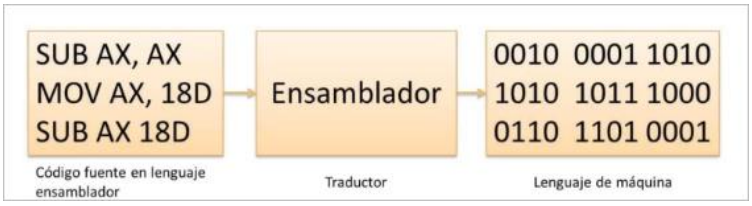
P. Ensambladores:
utilizan el lenguaje ensamblador. A partir de un programa fuente, creado por el programador, el P. ensamblador, crea un programa objeto. Cada instrucción en L. ensamblador es traducida a una instrucción en lenguaje de máquina, éste luego se ejecuta.

Ejemplos de lenguaje de máquina:

```
8B542408 83FA0077 06B80000 0000C383

FA027706 B8010000 00C353BB 01000000

B9010000 008D0419 83FA0376 078BD989
```

```
11001010 00010111 11110101 00101011
00010111 11110101 00101011 00101011
11001010 00010111 11110101 00101011
00010111 11110101 00101011 00101011
11001010 11110101 00101011 00101011
11001010 11001010 11110101 00101011
11001010 11110101 00101011 00101011
11001010 00010111 11110101 00101011
00010111 11110101 00101011 00101011
11001010 11110101 00101011 00101011
```

Ejemplo de lenguaje ensamblador y traductor:



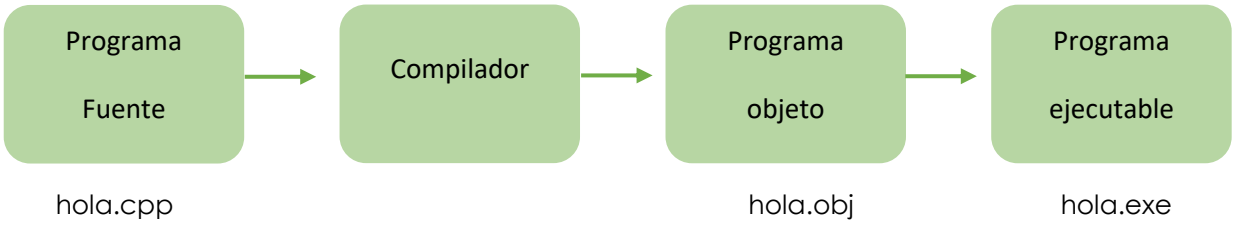
P. Intérprete: utilizan lenguaje de alto nivel, se ejecuta el programa fuente línea a línea. P.e.: javascript, java, php, python, perl, ruby, etc. En general, son lenguajes utilizados para aplicaciones web.

Ejemplos de lenguaje interpretado

```python
import random

def buscarElemento(lista, elemento):
    for i in range(0,len(lista)):
        if(lista[i] == elemento):
            return i

def imprimirLista(lista,nombre):
    for i in range(0,len(lista)):
        print nombre + "[" + str(i) + "]=" + str(lista[i])

def leerLista():
    lista=[]

    i=0
    while i < 10:
        lista.append(int(random.randint(0, 10)))
        i=i+1
    return lista

A=leerLista()
imprimirLista(A,"A")
cn=int(raw_input("Numero a buscar: "))
print "A[" + str(buscarElemento(A,cn)) + "]"
```

```php
<?PHP
    $mensaje_es="Hola";
    $mensaje_en="Hello";
    $idioma = "es";
    $mensaje = "mensaje_" . $idioma;
    print $$mensaje;
?>
```

P. Compilador: utilizan lenguaje de alto nivel. A partir de un programa fuente, creado por el programador, el compilador, crea un programa objeto, si no hay ningún error, se crea el ejecutable.



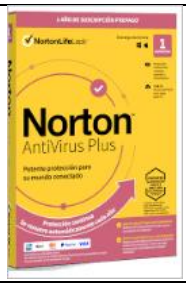
Ejemplos de lenguaje compilado

```c
#include <stdio.h>

int main (void){

    int opcion;
    float Euros, Pesetas;

    printf("\n 1: Euros a Pesetas");
    printf("\n 2: Pesetas a Euros\n");
    printf("\nEscoja la opcion deseada: ");
    scanf("%d", &opcion);

    if(opcion==1){
        printf("\n\n Introduzca la cantidad en Euros: ");
        scanf("%f", &Euros);
        printf("\n %f Euros son %f Pesetas", Euros, Euros*166.386);
    }else{
        printf("\n\n Introduzca la cantidad en Pesetas: ");
        scanf("%f", &Pesetas);
        printf("\n %f Pesetas son %f Euros", Pesetas, Pesetas/166.386);
    }
    return 0;
}
```

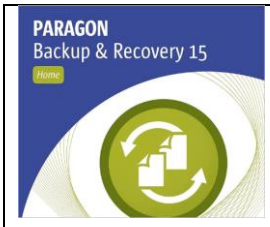
```java
public class Binario {
    public static void main(String[] args) {
        byte a=12;
        byte b=-12;
        byte c=6;

        System.out.println("12 >> 2 ="+(a >> 2));
        System.out.println("-12 >> 2 ="+(b >> 2));
        System.out.println("-12 >>> 2 ="+(b >>> 2));
        System.out.println("12 << 2 ="+(a << 2));
        System.out.println("-12 << 2 ="+(b << 2));
        System.out.println("12 & 6 ="+(a & c));
        System.out.println("12 | 6 ="+(a | c));
        System.out.println("12 ^ 6 ="+(a ^ c));
        System.out.println("~12 ="+~a);
    }
}
```

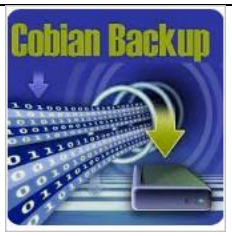
Software de Utilidad: programas que ayudan a la gestión del sistema

- Antivirus.



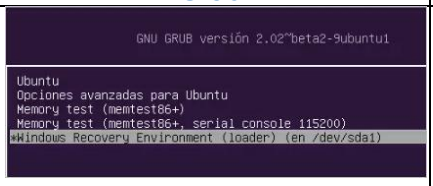
- Copias de Seguridad – Backups





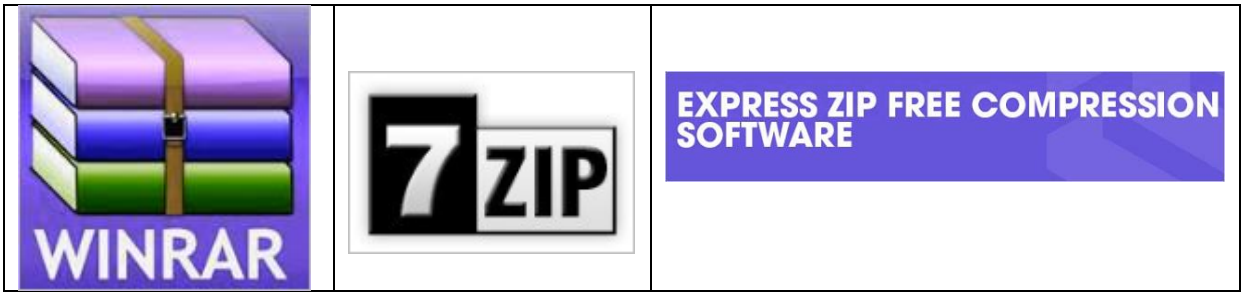
- Gestión de Discos

- Gestores de arranque

Bootit	Grub	Windows 10
		

- Utilidades de mantenimiento y diagnóstico

- Utilidades de compresión



- Utilidades de grabación de CD/DVD

- Utilidades de clonación.